Docket No. P2092D/1612US

APPLICATION FOR

UNITED STATES UTILITY LETTERS PATENT

Be it known that we, David V. James and William Rivard, have invented certain new and useful improvements in an

APPARATUS AND METHOD FOR INTER-NODE COMMUNICATION

of which the following is the Specification:

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail No. EK889946752US in an envelope addressed to: BOX PATENT APPLICATION, Commissioner of PatentS, Washington, D.C. 20231, on December 31, 2001.

Nancy R. Simon

(Signature)

CROSS REFERENCE TO RELATED APPLICATION

This is a division of United States Patent Application No. 09/040,149, filed on March 17, 1998.

BACKGROUND

1. Technical Field

The present invention relates generally to apparatus and methods for transmitting signals between nodes, and more particularly for transmitting signals at high bit-rates between nodes.

2. Background

A typical computer or inter-linked set of computers can be modeled as a series of nodes which communicate with one another point-to-point. Although nodes have in the past been attached to a bus, modern communications standards more commonly employ point-to-point interconnections. In the past, the communication data rate between such nodes was limited more by the performance of the computer or its various internal chips than by the speed of the traces or transmission lines by which the nodes were connected. Now as chip and computer speeds have substantially increased, these interconnects are hindering further performance improvements.

More specifically, current signaling architectures as well as the physical limitations of the traces and transmission lines themselves limit the maximum inter-node communications rate. Synchronized buses and point-to-point links are two of interconnect architectures commonly used. Synchronous bus architectures typically broadcast an address block to all nodes on a multiplexed bus. The node corresponding to the address then generates an acknowledgement block, which is also broadcast to all of the nodes. This architecture results in relatively low communications

throughput. This is because each node must be synchronized to the same common reference clock so that address and acknowledgment blocks can be transmitted and received. Nodes employing the synchronous bus architecture must also take turns communicating, further limiting the maximum possible inter-node data rate, especially when a large number of nodes are connected to the same bus. The multiplexed buses used with synchronous bus architectures also typically contain intermediate stubs and additional signal paths that limit the effective speed of data transfer between nodes.

Point-to-point link architectures are comparatively more time efficient since their signals have an affiliated reference clock signal. Using a point-to-point link architecture, two nodes can transfer data at a rate independent of other nodes and of any common reference clock. However, point-to-point link inter-node data rates still tend to be limited by the physical limitations in the traces and transmission lines.

For instance, modern computers typically communicate by either single- ended or differential signaling. Both of these forms of signaling are well known in the art. Ideally, single-ended connections require only one physical line per logic signal. However, as communication rates have increased so has ground-bounce, which is inherent in single-ended systems. Attempts to solve the ground-bounce problem include adding power supply and ground pins for each single-ended logic line on a chip, effectively tripling the number of physical traces required. Thus, six single-ended logic signals can require up to eighteen physical traces. Differential signaling systems require two physical traces for each logic signal. Thus, six differential logic signals require at least twelve physical traces. Since silicon and computer resources are finite, a large number of traces or transmission lines can significantly increase the cost of manufacturing the chip or the computer.

Regardless of whether single-ended or differential signaling is used, the physical traces and transmission lines all have an inherent parasitic inductance. As the data rate over these pathways increases, the parasitic inductance combined with the quickly varying signal currents generate parasitic voltages that interfere with and corrupt the signals traveling over these pathways.

Additionally, large signal currents that pass through the traces and transmission lines can generate Electro-Magnetic Interference (EMI) noise which further corrupts signals traveling between the nodes. Such EMI noise may also, from time to time, exceed the limits of various well known regulatory standards for permissible EMI radiation levels.

Other prior art approaches have employed RAMBUS technology (manufactured by RAMBUS, Inc. of Mountain View, California) to reduce the parasitic and EMI noise voltages present on some signal lines. The RAMBUS approach consists of a number of traces or transmission lines, each of which transmits a different signal. Ideally, these signal lines are kept in close proximity to one another. One of the signal lines is designated as a reference and used to cancel out some of the noise effects present on the signal lines. A shortcoming of this approach is a noticeable current surge when all of the signal lines are either logic 1's or logic 0's.

SUMMARY

The present invention delineates an inter-node communications paradigm for enabling signals to be transmitted between nodes at a higher rate. The higher rate is possible due to an encoding schema that reduces current demands and fluctuations between multiple nodes. The encoding schema also requires fewer physical traces and/or transmission lines than high speed single-ended and differential signaling circuits.

Within the apparatus of the present invention, a first node is connected to a communication channel. Operations on the first node result in a first set of signals that are to be transmitted over the communication channel. The logic states which comprise the first set of signals may range from all logic zeros to all logic ones. This large number of potential logic transitions results in large current fluctuations over the communication channel. To reduce and/or eliminate these current fluctuations, the present invention also includes an encoder or lookup table for transforming the first set of signals into a second set of signals having either an equal number, nearly an equal number, a constant number, or nearly a constant number of logic ones and logic zeros. In one embodiment, groups of six signals from the first set of signals are encoded into eight signals in the second set of signals.

Within the method of the present invention, a first set of signals from a first node are encoded into a second set of signals having either an equal number, nearly an equal number, a constant number, or nearly a constant number of logic ones and logic zeros. This second set of signals is then transmitted over a communication channel.

Thus, the present invention presents a communications technique which has quieter switching currents than single-ended circuits and requires fewer physical traces and transmission lines than differential circuits. The

present invention can be applied to communications between computer chips on a circuit board as well as between nearby computers linked together. These and other aspects of the invention will be recognized by those skilled in the art upon review of the detailed description, drawings, and claims set forth below.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of an apparatus for inter-node communication;

Figure 2 is a table of unencoded signals utilized by each node within the apparatus of **Figure 1**;

Figure 3 is a block diagram of an apparatus for encoding and transmitting signals between a set of nodes;

Figure 4 is a partial electrical circuit of an encoder and decoder pair within the apparatus for encoding and transmitting the signals between the set of nodes;

Figure 5 is a flowchart of a method for inter-node communication; and

Figure 6 is a flowchart of a method for encoding inter-node communication signals.

DETAILED DESCRIPTION

Figure 1 is a block diagram of an apparatus 100 for transmitting high bit-rate signals between a set of nodes. The apparatus 100 includes a first node 102, a second node 104, a third node 106, a fourth node 108 and a clock generator 110. The apparatus is designed to provide the functionality of a synchronous bus, with the performance advantages of point-to-point signaling. Each node 102, 104, 106 and 108 receives a clock signal on line 112 from the clock generator 110, data signals on lines 114 A, B, C and D, flag signals on lines 116 A, B, C and D, and strobe signals on lines 118 A, B, C and D. Collectively, the signals on lines 114, 116, and 118 make up a shared communications channel/link between the nodes 102, 104, 106 and 108. Those skilled in the art will be aware of various other receive and transmit signals which can also be passed between the nodes 102, 104, 106 and 108, depending upon the application or transmission protocol.

Data transmissions between nodes 102, 104, 106 and 108 are source synchronous. Even though all of the nodes also receive the clock signal on line 112, the strobe signals on lines 118 A, B, C and D act as a reference clock for these data transmissions. The strobe signal resolves any frequency differences in timing between the transmitting and receiving node. Although each node has to compensate for arbitrary frequency differences between its incoming data and the clock signal on line 112, this arrangement eliminates the need to insert or delete symbols to adjust for inter-node timing differences. Strobe signals also enable higher transmission bandwidths between the nodes because there is no longer a need to accurately synchronize all of the nodes within the apparatus 100. The inter-node transmission rates are not affected by inter-node transmission delays, thus arbitrary delays between each of the nodes are acceptable.

Using the point-to-point link data communication architecture, the nodes transmit requests and responses as *send* packets. The receiving node strips the *send* packet of its data and then returns a small *acknowledgment* (i.e. *"ack")* packet. Routing architectures for the packets are preferably straightforward. In the preferred embodiment, the *send* packets addressed to a specific node are stripped from the inter-node data stream flowing on lines 114 A, B, C, and D by that node and are replaced with *ack* packets. The *send* and *ack* packets addressed to other nodes are passed through any intermediate nodes. The *ack* packets are stripped from the inter-node data stream when they return to the original transmitting node.

For example, in order for node 102 to transmit data to node 108, node 102 must first generate a *send* packet addressed to node 108. Node 102 then transmits this *send* packet to node 106. Node 106 looks at the address of the *send* packet and since the *send* packet is not intended for node 106, passes the *send* packet along to node 108. Node 108 then looks at the address of the *send* packet and since the *send* packet is intended for node 108, strips the *send* packet from the inter-node data stream and generates an *ack* packet addressed to node 102. Node 108 then sends the *ack* packet to node 104. Node 104 looks at the address of the *ack* packet and since the *ack* packet is not intended for node 104, passes the *ack* packet along to node 102. Node 102 then looks at the address of the *ack* packet and since the *ack* packet is intended for node 102, strips the ack packet from the inter-node data stream. After this last step the data exchange between nodes 102 and 108 is complete.

The flags on lines 116 A, B, C, and D are used for data stream packet-framing. Data stream packet-framing consists of labeling each packet with either a first-packet symbol, a between-packet-idle symbol, or a last-packet symbol. These flags are also used for arbitration purposes. Arbitration protocols place a limit on a number of consecutive packets that can be sent by

any one node. This ensures that each node has an opportunity to send its packets over the shared link.

Preferably one, and only one, of the nodes 102, 104, 106, 108 functions as a "scrubber" within the apparatus 100. The scrubber performs such maintenance functions as, removing mis-addressed packets on their second pass around the shared link. The scrubber can be selected by setting a *scrubId* line 120 A, B, C or D on the selected node to logic "1" (node 104 in Figure 1) and setting the remaining *scrubId* lines 120 A, B, C or D to logic "0" (nodes 102, 106 and 108 in Figure 1). Those skilled in the art will know of other vendor-dependent scrubber-assignment techniques that can be used.

Figure 2 is a table 200 of unencoded signals utilized by each node within the apparatus of **Figure 1**. The signals include a 1-pin scrubber identifier (scrubId) 202, a 1-pin central clock (clock) 204, a 1-pin strobe 206, 4-pins of flags 208, and 32-pins of data.

The scrubId signal 202 is received by the nodes on lines 120 A, B, C, and D, and uniquely identifies a particular node as the scrubber within the shared communications channel (also known by those skilled in the art as a ringlet). In an alternate embodiment, a vendor-dependent scrubber-identification technique can be provided.

The clock signal 204 is preferably an input-only signal received by the nodes on line 112, and provides the nodes with a reference for synchronizing their internal frequency-lock loops. The frequency-lock-loops within the nodes permit a lower clock generator 110 frequency, which in turn simplifies clock signal distribution. For example, a 50 MHz clock signal can be frequency-lock-looped up to 500 MHz. Also since the frequency-lock loops within the nodes are tracking the clock signal fairly closely, the cycles per second each node sees is about the same.

Frequency locking, rather than phase locking, is used because arbitrary, but fixed, phase differences can be tolerated and a small input-data FIFO can compensate for any incoming phase differences. The FIFO compensates for the fixed phase error between each node. The fixed phase error is random and cannot be controlled.

The strobe signal 206, received by the nodes on lines 118 A, B, C, and D, is preferably complemented on each cycle, so that a receiving node can accurately determine when incoming (source-synchronous) data should be latched.

The flag signals 208, received by the nodes on lines 116 A, B, C, and D, transfer control and framing information between the nodes.

The data signals 210, received by the nodes on lines 114 A, B, C, and D, transmit the contents of the *send* and *ack* packets. Preferably, most-significant bits are sent first when sending a packet header, and lower addresses are sent first when sending data. While the present inter-node data packets contain 32 data bits, other data packet capacities, such as 8, 16, 64, or 128 data bits, are also acceptable. Alternatively, large data-words may be broken up and sent over multiple clock cycles. For instance a large 64 bit data packet could be sent as two smaller 32 bit data packets. When deciding how many bits to include in a data packet, designers should consider that while 8-bit data packets may be more cost effective in terms of coding or hardware to implement, such small data packet designs have less bandwidth per pin due to the relatively-fixed overhead of the scrubId, clock, strobe, and flag pins. However, while 128-bit data packets may have more bandwidth per pin, such large data packets may be more expensive to implement in terms of coding, hardware, and/or skew-management circuits.

Figure 3 is a block diagram of an apparatus 300 for encoding and transmitting the signals 206, 208, and 210 between a set of nodes. The apparatus includes a driver node 302 and a receiver node 304 which can represent any pair of the **Figure 1** nodes 102, 104, 106, and 108. The driver node 302 includes a plurality of encoders 302 A-G and the receiver node 304 includes a plurality of decoders 304 A-G. The nodes 302 and 304 communicate via a set of traces, circuit paths, and/or transmission lines 308. These traces, circuit paths, and transmission lines 308 collectively make up part of the communications channel between the two nodes and perform the same roles as the data 114, flag 116, and strobe 118 lines of **Figure 1**. The nodes themselves can include any number of inter-linked devices. These inter-linked devices can be computer chips, circuit boards, and/or stand alone computers.

The apparatus 300 transmits the signals between the nodes by either a dedicated line (such as for the scrubId signals on lines 120 A-D and the clock signal on line 112) or after implementing an encoding schema (such as for the strobe, flags and data signals). By selectively encoding the strobe, flag and data signals transmitted between the nodes 302, 304, ground-bounce during high-speed signaling, can be reduced. Preferably the strobe, flag and data signals are grouped and encoded in such a way that nearly an equal (called DC-free encoding) and/or constant (called DC-balanced encoding) number of logic 0's and l's are always transmitted between the driver node 302 and the receiver node 304. These encoding schemas are preferably implemented using an even number of data lines 308.

One way to implement a DC-balanced encoding schema is shown in **Figure 3**. The strobe 206 signal is fed into a 1 to 2 encoder 302A which uses a complementary encoding schema before the signal is transmitted over the data lines 308. The flag 208 and data 210 signals, however, are divided into groups of six unencoded signals (i.e. 6-bits), which are then converted into

groups of eight encoded signals (i.e. 8-bits). These eight encoded signals are then transmitted in parallel over the data lines 308. Other signal coding schemas, such as when groups of four unencoded signals (i.e. 4-bits) are converted into groups of six encoded signals (i.e. 6-bits), can also be used.

DC-free encoding schemas can be implemented by transmitting an even number of encoded signals with an equal number of logic 1's and 0's in parallel over the data lines 308.

The DC-Balanced and DC-Free encoding schemas have the following characteristics: First, a constant number of 1's valued lines is always driven and thus the driver node 302 is balanced. A balanced driver node means that total current over the data lines 308 is fairly constant. Second, ground-bounce noise is reduced, since logic transitions from all zeros to all ones and from all ones to all zeros are eliminated. Third, an implied reference voltage can be obtained by averaging all of the data line 308 voltages. Fourth, parity protection is inherent, since all single-bit transmissions failures, and many double-bit errors, can be detected as an illegal (i.e. non-DC-balanced) input code value. Fifth, peak current demands are reduced, since for each of the unencoded signals, only half of the data lines are actively driven to a high current logic state (such as logic "1"). Sixth, the encoding schema allows extra control characters to be transmitted (for instance, in a 6-to-8 bit encoding schema there are 6 unmapped 8-bit encoded values for each set of 64 unencoded values).

A nearly DC-free/nearly DC-balanced encoding schema can be implemented by transmitting an odd number of bits, which contain no more than one extra logic 1 or logic 0, in parallel over the data lines 308. For instance, a 6/7 encoding schema (i.e. where 6-bits are encoded into 7-bits) may be used where a logic 1 to logic 0 ratio is either 3-to-4 or 4-to-3. Although more efficient, this nearly free/balanced encoding schema has no parity

protection and may be subject to signal-integrity limitations. This encoding schema also results in a less accurate threshold reference voltage, once all of the received signal values are averaged by the decoders 304 A-G.

The encoding schema herein taught requires extra-pinouts (for example, the 6-to-8 schema requires an additional 2-pins for every 6 signals). However, these additional demands upon limited silicon and PC-board resources are offset by a reduction in a number of required ground and/or power pins when compared with current unencoded full-swing signals transmitted over single-ended data lines. To localize any chip or PC-board ground-plane currents, each set of 8 signal traces is preferably routed as a group. While the 6-to-8 bit encoding schema is preferred, sometimes the number of unencoded signal lines are not always multiples of 6. In those cases other encoding options are possible, such as 1-to-2 (differential), 2-to-4 (also differential), and 4-to-6 encoding schemas.

Figure 4 is a partial electrical circuit 400 of the encoder 302A and the decoder 304A pair within the apparatus 300 for encoding and transmitting the signals between the set of nodes. The circuit 400, shown in **Figure 4**, effects a 6-to-8 bit encoding schema which employs a simple technology-independent technique for driving the data lines 308 by switching a constant current 402 (i.e. "4i") to half (i.e. 4) of the total number (i.e. 8) of data lines 308 which are driven to a high current logic state lines. Only half of the data lines are driven since a DC-free encoding schema is effected. A zero volt termination voltage 404, rather than a fixed positive voltage, enables the circuit to be supply-voltage independent. In the circuit 400, termination resistors, R_t, are effected using on chip FETs. The value of R_t is matched to the impedance of each data-line 308. A typical termination resistance is 75Ω. Averaging resistors, R_c, are chosen so that the threshold reference voltage of the decoder 304A is one half of the driven signal levels on the data lines 308. The decoder 304A then uses differential amplifiers 406 to compare the signals on the

data-lines 308 with the threshold reference voltage to determine whether, for example, a logic "1" or a logic "0" has been transmitted.

Portions of the encoding and decoding hardware described in **Figure 3** may also be implemented in software. In so doing, additional configurable elements, such as a processing unit, a memory, and a storage device (not shown) need to be added to **Figures 1, 3, and 4**. The memory would store computer program instructions for controlling how the processing unit accesses, transforms and outputs data. Those skilled in the art will recognize that in alternate embodiments the memory could be replaced with a functionally equivalent computer-useable medium such as a compact disk, a hard drive or a memory card.

Figure 5 is a flowchart of a method for inter-node communication. The method begins in step 502 where a driver node and a receiver node are identified within a communications system. Next, in step 504, a set of unencoded signals transmitted from the driver node to the receiver node are identified. The signals include a set of data values (i.e. logic "1's" and logic "0's"). In step 506, the unencoded data values of the unencoded signals are encoded to produce encoded signals. Next, in step 508, the encoded signals are transmitted from the driver node to the receiver node. In step 510, the encoded signals are decoded. After step 510, the method for inter-node communication is complete.

Figure 6 is a flowchart of a method for encoding the data values of the inter-node communication signals (step 506 of **Figure 5**). The method begins in step 602 by selecting a code such that a difference between a total number of unencoded data values and a total number of encoded data values is a small predetermined fraction of the total number of unencoded data values. A 6-bit unencoded signal to 8-bit encoded signal coding schema is preferred, however a 4-bit unencoded signal to 6-bit encoded signal coding schema may be more

appropriate in some applications. Codes may be stored in a set of lookup tables or generated using algorithmic methods. Codes may also be switched at any point during signal transmission provided that the switch is communicated to the receiver node. In step 604, the code is selected such that an encoded signal has an equal number of logic 1's and 0's. Alternatively, in step 606 the code is selected such that an encoded signal has nearly an equal number of logic 1's and 0's. Alternatively, in step 608 the code is selected such that an encoded signal has a constant number of logic 1's and 0's. Alternatively, in step 610 the code is selected such that an encoded signal has nearly a constant number of logic 1's and 0's. After step 610, the method for encoding the inter-node communication signals is complete.

While the present invention has been described with reference to a preferred embodiment, those skilled in the art will recognize that various modifications may be made. Variations upon and modifications to the preferred embodiment are provided by the present invention, which is limited only by the following claims.